Does Process Assessment Drive Process Learning? The Case of a Bachelor Capstone Project

Vincent Leilde, Vincent Ribaud

Does process assessment drive process learning?

The case of a Bachelor capstone project

Vincent Leildé
Lab-STICC, team MOCS, ENSTA Bretagne
Brest, France
v.leilde@gmail.com

Vincent Ribaud
Lab-STICC, team MOCS, UBO
Brest, France
ribaud@univ-brest.fr

***Abstract* — In order to see if process assessment drives processes learning, process assessments were performed in the capstone project of a Bachelor in Computer Science. Assessments use an ability model based on a small subset of ISO/IEC 15504 processes, its main Base Practices and Work Products. Students' point of view was also collected through an anonymous questionnaire. Self-assessment using a competency model helps students to recognize knowledge, skills and experience gained over time and in diverse contexts. The capstone project offered a starting point. Students' self-assessment and external assessment are correlated to some point but are not correlated for topics unaddressed in the curriculum or unknown by students.**

Keywords—process assessment; ability model; capstone project

I. INTRODUCTION

At Brest University, the Bachelor capstone lasts roughly 100 hours during the Bachelor last semester. Students learn by doing a simplified cycle of software development through a somewhat realistic project. A rainbow metaphor is used to position phases in the cycle, divided in four main phases: requirement (*red*), design (*yellow*), construction (*blue*) and integration (*indigo*). Three phases are used to transition between main phases: requirements analysis (*orange*), detailed design (*green*), validation (*violet*). A side-effect goal of the capstone project is to be exposed to some kind of process assessment. A process is "a set of interrelated or interacting activities which transforms inputs into outputs [1]." Process assessment is applicable by or on behalf of an organization with the objective of understanding the state of its own processes for process improvement. Our hypothesis is that it is worth it for students, if they are conscious of the processes underlying their way-of-working.

It is a common assertion that assessment drives learning. Our research question examines whether process assessment drives process learning. 25 students consented to use an ability model based on a small subset of the ISO/IEC 12207:2008 Process Model [1] and self-assessed skills proficiency test using a 4-point Likert scale. Teachers assessed students using the same ability model and Likert scale, and for two processes, a 3rd-party assessment was also performed. Volunteer collected the results including a satisfaction questionnaire, gathered students' and teachers' assessment and anonymized data. Results are used to present the achievement degree of students together with a possible correlation between self and external assessments. Some initial findings were presented in [2]. The structure of the paper is: section 2 overviews process assessment; section 3 presents different assessments we carried out; section 4 presents the questionnaire results and we finish with a discussion and a conclusion.

II. PROCESS ASSESSMENT

A. ISO/IEC 15504 standard

ISO/IEC 15504 standard uses a Process Assessment Model which is a two-dimensional model of process capability. In the process dimension, processes are defined and classified into process categories. In the capability dimension, a set of process attributes grouped into capability levels is defined. Process Attributes are features of a process that can be evaluated on a scale of achievement, providing a measure of the capability of the process. The process dimension uses a Process Reference Model (PRM) replicated from ISO/IEC 12207:2008. A PRM is a model comprising definitions of processes in a life cycle described in terms of process purpose and outcomes, together with an architecture describing the relationships between processes. The capability dimension defines process assessment that is essentially a measurement activity of process capability on the basis of evidences related to assessment indicators. There are two types of assessment indicators: process capability indicators, which apply to all capability levels and process performance indicators, which apply exclusively to capability level 1. The process performance indicators are Base Practices (BP) and Work Products (WP). "A base practice is an activity that, when consistently performed, contributes to achieving a specific process purpose", and "a work product is an artifact associated with the execution of a process" [1].

B. Ability Model

Computer Science bachelor (CS) students are generally focused either on technical topics or theoretical subjects. Little attention is paid to software engineering in a CS Bachelor curriculum. The PRM we use is a small subset of the ISO/IEC 15504:2006 Process Reference Model, mainly the Software-related Processes of the ENG Process. Process Purpose, Process Objectives and Base Practices have been kept without any modification; Input and Outputs Work Products have been reduced to main products.

To foster a practical understanding of SE, we use a colored cycle and we slightly rearrange the ENG Software-related Processes Process Group. The cycle is ENG.1 Requirements elicitation: *red*; ENG.2 System requirements analysis: *orange*;

ENG.3 System architectural design: *yellow*; ENG.5 Software design: *green*; ENG.6 Software construction: *blue*; ENG.7 Software integration: *indigo*; ENG.8 Software testing: *violet*. From an individual human perspective, this subset can be seen as a competency model related to the knowledge, skills and attitudes involved in a software development project. A competency model (also called an ability model) defines and organizes the elements of a curriculum and their relationships.

A hierarchical model is easier to manage and use. We kept the hierarchical decomposition issued from the 15504: process groups – process – base practices and products. A competency model is decomposed into competency areas (mapping to process groups); each area roughly corresponding to one of the main division of the profession or of a curriculum. Each area organizes the competencies into families (mapping to processes). A family roughly corresponds to main activities of the area. Each family is made of a set of knowledge and abilities (mapping to base practices), eventually called competencies; each of these entities being represented by a designation and a detailed description. The ability model and its associated tool *eCompas* has been presented in [3].

C. Process Assessment

ISO 15504 [1] defines a measurement framework for the assessment of process capability defined on a six point ordinal scale which represents increasing capability of the implemented process, from not achieving the process purpose through to meeting current and projected business goals [1]. Capability Level 0 denotes an incomplete process, either not performed at all, or for which there is little or no evidence of systematic achievement of the process purpose [1]. Capability Level 1 denotes a performed process that achieves its process purpose through the performance of necessary actions and the presence of appropriate input and output work products which, collectively, ensure that the process purpose is achieved [1]. We are not interested in higher levels than 1 in this study. Evidence of performance of the base practices, and the presence of work products with their expected work product characteristics, provide objective evidence of the achievement of the purpose of the process. Achievement is characterized on a defined rating scale: N Not Achieved, P Partially Achieved, L Largely Achieved, F Fully Achieved.

If students are able to perform a process, it denotes a successful learning of software processes, and teachers' assessments rate this capability. The research question aims to state if self-assessment and external assessment help students to improve their software processes practice and how students, teacher and third-party assessment are correlated.

III. The Capstone Project

A. Overview

1) Schedule

The curriculum is a 3-year Bachelor of Computer Science. The project happens the third year before students' internship. The project is performed during a dedicated period of two weeks. Before the dedicated weeks, a dozen of two-hour labs are conducted all the semester along and some homework is required. According to students' estimates, they spent an average of 102 hours on the capstone project. Each phase is driven by main Base Practices of the related software process and ends up with the delivery of few related Work Products. Each deliverable can be reviewed as much as needed by the teacher that provides students with comments and suggestions. When the dedicated period starts, students are familiar with the Author-Reader cycle and have performed the requirements and architectural design processes.

2) Statement of work

The software is made of 2 sub-systems: PocketAgenda that manages an address book and an agenda and interfaces with a central directory; WhoIsWho manages the directory and a social network. PocketAgenda is implemented with Java and JSF relying on an Oracle RDBMS. WhoIsWho is implemented in Java using a small RDBMS. Both sub-systems interact in a client-server mode and communicate with a protocol established using UDP.

3) Students' consent

Students were advised that they can freely participate in the experiment described in this paper. The class contains 29 students, all agreed to participate; 4 did not complete the project and do not take part to the study. Students have to regularly update the competency model consisting of the ENG process group, the 6 processes above and their Base Practices and main Work Products and self-assess on the N-P-L-F scale. There are also teacher and 3rd-party assessments that have to be attached to self-assessments by volunteer students.

4) Statistics

Process assessment was continuous and communicated to students regularly; hence they were made aware of their progression very often and adjusted their effort. Table 1 presents teacher's assessment. BP and WP rating are aggregated using an all-or-none principle: if all BPs or WPs in a process are rated at least Not, Partially, Largely or Fully, BPs or WPs are rated Not, Partially, Largely or Fully.

TABLE I. TEACHER'S ASSESSMENTS (AGGREGATED)

	Base Practices				Work Products			
Rating	N	P	L	F	N	P	L	F
ENG.1/2 Requirement	0	5	19	1	0	5	17	3
ENG.3/5 Design	0	8	17	0	0	5	17	3
ENG.6 Construction	0	4	21	0	17	7	1	0
ENG.7 Integration	0	5	12	8	1	1	17	6
ENG.8 Testing	3	19	3	0	25	0	0	0

B. Writing and analyzing requirements

Before the 2-weeks dedicated period starts, students have to capture, write and manage requirements through use cases. It is a non-technical task unfamiliar to students. Students report that eliciting and writing requirements were a difficult task and the Author-Reader cycle helped to produce complete and usable use cases and to acquire a writing style. In most cases, three cycles were required to achieve the task. According to students' estimates average, they spent 20 hours (roughly a fifth of the total hours) to capture, write and

manage use cases. This period refers to *red/orange* colors corresponding to the ENG.1 and ENG.2 processes. A 4-hour lecture about use cases was delivered in January at the beginning of the semester, then the iterative process of writing and being reviewed by the teacher started.

TABLE II. ENG.1/2 ASSESSMENT (SELF AND TEACHER)

	Stud.	***Teach.***	***r***
BP1: Specify software requirements	2.12	1.84	0.31
BP3: Develop criteria for software testing	1.76	1.76	1.00
BP4: Ensure consistency	1.92	0.88	0.29
17-8 Interface requirements	1.88	1.88	1.00
17-11 Software requirements	2.08	2.08	1.00

Table 2 presents the correlation coefficient `r` between student and teacher assessment for the ENG.1 Requirements elicitation and ENG.2 Requirements analysis processes. It relies on 3 Base Practices and 2 Work Products. Table 2 presents also the average assessment for each assessed item. The correlation coefficient of each item relates 25 self-assessment measures with the corresponding teacher assessment measures; the overall correlation coefficient relates 25 * 5 = 125 couple of measures, its value `r = 0.64` indicates a correlation.

Thanks to the Author-Reader cycle, the final mark given to almost all Interface requirements and Software requirement documents was Largely Achieved or Fully Achieved. Students were made aware of their mark and reproduced the mark during the self-assessment; obviously the correlation between students and teacher assessments is complete. However, students mistook documents assessment for practices assessment. Documents were improved through the author-reader cycle, but only reflective students improve their practices accordingly. Other students self-assess their practices at the same level that the corresponding work products, but the teacher did not; see for example BP1: Specify software requirements. Also, students failed the self-assessment for BP4: Ensure consistency. Most students neglected traceability and self-assessed at a higher level that the teacher did.

An abnormal set of values can bias a correlation coefficient; if we remove the BP4: Ensure consistency assessment, we get `r = 0.89`, indicating an effective correlation. However, a bias still exists because students are assessing themselves using the continuous feedback they got from teachers during the Author-Reader cycle.

C. Architecture and integration: a client-server endeavor

The ACM/IEEE joint Computer Science Curriculum [4] states about the capstone project "*Such projects should challenge students by […] requiring work on a larger scale than typical course projects. Students should have opportunities to develop their interpersonal communication skills as part of their project experience.*" The first week was used for these purposes: students have to work closely in pairs, to produce interface specification and to integrate the client and server sub-systems, each sub-system being designed, developed and tested by one student. The scope of the week is bounded and defined by 4 use cases, previously written by students. The architectural design has been already done during the semester using SADT and E-R models. The week schedule follows a kind of agile sprint schedule: agreement on requirements and high-level architecture (0.5 day), pair working interface design and low-level decisions (1-1.5 days), individual sub-systems development and test unit (2.5-3 days). Some pairs chose continuous integration, some other pairs performed integration the last day of the week. Both scheme worked but 4 pairs failed to work together and split, consequently lone students worked alone and have to develop both sub-systems, with or without success.

We designed the assessment with the intent to minimize the bias mentioned in the previous section: students interpret the teacher's feedback to self-assess accordingly. Also, we wished to focus on the pair work and the architectural design / integration relationship. Hence, we focus the first week assessment on ENG.3 System architectural design process (*yellow*) and ENG.7 Software integration process (*indigo*). One author worked several years in a software company and had some experience in 15504 assessments, and then he did not participate to the week and acted as a third-party assessor. The other author and another teacher coached and assessed students' BPs and WPs during the whole week.

1) Architectural design

UML modeling and object-oriented design are taught in dedicated lectures (30 hours each). However, nearly all students had no idea how to perform architecture and interface design. Architectural design was taught by example: teachers performed a step-by-step design for one of the four use cases; students reproduced the scheme for the remaining use cases.

TABLE III. ENG.3 (SELF, TEACHER AND THIRD-PARTY)

	Std. avg	***Tch. avg***	***3rd avg***	***r Std-Tch***	***r Tch-3rd***
BP1: Describe system architect.	2.24	2.02	1.68	-0.22	0.18
BP3. Define interfaces	1.96	2.16	1.56	0.48	0.36
BP4. Ensure consistency	2	1.72	0.88	0	0.44
04-01 Database design	2.48	2.2	1.88	0.49	0.35
04-04 High level design	2.12	1.84	1.64	0.37	-0.11

For the ENG.3 System architectural design, Table 3 presents the correlation coefficient between student and teacher assessments and the correlation coefficient between student and third-party assessments. Assessment relies on 3 Base Practices and 2 Work Products. Table 3 presents also the average assessment for each assessed item. The overall correlation coefficient between self-assessment and teacher assessment measures is `r1 = 0.28` and the overall correlation coefficient between self-assessment and third-party assessment measures is `r2 = 0.24`. There are no real differences and indeed no indication for a correlation.

In Table 3, we see that item correlation is poor, except for database design and interface design, but these topics are deeply addressed in the curriculum. An half of students performed a superficial architectural work because they are

eager to jump to the code. They believe that the work is fair enough and teachers as well, the external assessor does not. The BP4.Ensure consistency is a traceability matter suffering the same problem described above. Teachers also have a weak awareness of the topic; they over-assess students; and there is no correlation with students' and teachers' assessments.

Students reported that requirement analysis with SADT greatly helped to figure out the system behavior and facilitated the design phase and interface specification. However, students had never really learnt architectural design and interface between sub-systems, indeed it explains the lower 3rd-party assessment average for BPS and WPs.

*2) **Integration***

The integration topic is not addressed in the Bachelor curriculum. In the best case, students respected their interface specifications and few problems arose when they integrated client and server code. In some cases, they were unable to perform the integration and the integration merely failed.

TABLE IV. ENG.7 AVERAGE (SELF, TEACHER AND THIRD-PARTY)

	Stnd. avg.	***Teac. avg.***	***3rd-party avg.***
BP1: Develop software integration strategy	1.56	1.20	0.40
BP2: Develop tests for integrated software items	2.08	1.08	0.52
BP3: Integrate software item	2.00	2.12	1.76
BP4: Test integrated software items	2.00	1.80	1.16
BP5. Ensure consistency	1.76	1.20	0.72
BP6: Regression test integrated software items	1.64	0.52	0.2
08-10 Software integration test plan	1.44	0.88	0.00
11-01 Software product	2.04	2.12	1.48

ENG.7 Software integration (*indigo*) is assessed with 6 Base Practices and 2 Work Products. The overall correlation coefficient between self and teacher assessments is `r1 = -0.03` and the overall correlation coefficient between self and 3rd-party assessments is `r2 = 0.31`. However, several BPs or WPs were assessed by the 3rd-party assessor with the same mark for all students (N or P): the standard deviation is zero and the correlation coefficient is biased and is not used. Table 4 presents the different assessments average.

All BPs and WPs related to integration and test are weakly third-party assessed, indicating that students are not really aware of these topics, a common hole in a Bachelor curriculum. Some students were aware of the poor maturity of the integrated product, partly due to the lack of testing.

D. Construction : information system development

The second week was devoted to perform a continuous and significant development endeavor. Students have to work loosely in pairs; each of them developed separate components of the information system and has been assessed individually. Unfortunately, a teacher quit the capstone project for strong personal reasons; consequently the 3rd party assessor moved back to be a teacher and an internal assessor.

1) Construction

JDeveloper is a Java IDE for the Oracle Application Development Framework (ADF). ADF is an end-to-end development framework, built on top of the Enterprise Java platform, and providing integrated solutions including data access, business services development, a controller layer, a JSF tag library implementation. Most of the application logic relies on the database schema, without the need to write code. During the semester, 16 labs hour were devoted to learning the framework, a few for mastering the IDE but enough for a start.

Java, database, network and SQL programming are taught in dedicated lectures during the curriculum (60 hours each). Despite of this amount, a third of students self-judged as having a poor knowledge of SQL and Java. Students have almost no idea of test-driven development and a lack of a test strategy; hence units were poorly tested. Although the Junit framework has been taught during the first semester, no student used it. These points have to be improved in the future.

TABLE V. ENG.6 ASSESSMENT (SELF AND TEACHER)

	Stud.	***Teach.***	***r***
BP1: Develop unit verification procedures	1.84	0.40	0.05
BP2: Develop software units	1.92	1.84	0.37
BP3: Ensure consistency	1.92	0.92	0.25
BP4: Verify software units	1.96	1.00	-0.2
17-14 Test cases specification	1.80	0.36	0.07
15-10 Test incidents report	1.52	0.12	-0.45

Table 5 presents the correlation coefficient `r` between student and teacher assessment for the ENG.6 Software construction process (*blue*). It relies on 4 Base Practices and 2 Work Products. Table 5 presents also the average assessment for each assessed item. The overall correlation coefficient is `r = 0.10` and there is no indication for a correlation.

Our bachelor students have little awareness of the importance of testing, including test specification and bugs reporting. Consequently, BPs and WPs related to unit testing were assessed by teachers with almost the same mark for all students (N or P), biasing the correlation coefficient. If we remove BPs and WPs related to unit testing management (17-14 Test cases specification; 15-10 Test incidents report; BP1: Develop unit verification procedures), we get `r = 0.49`, indicating a plausible correlation.

Students reported that the ENG.6 Software construction process raised a certain anxiety because students had doubt about their ability to develop a stand-alone server interoperating with a JDeveloper application and two databases but most students succeeded. For some students, a poor Java literacy compromised the project progress.

2) Qualification testing

On average, students spent less than 10% of the total hours to perform integration and qualification tests of the software. These topics are unaddressed in the curriculum and because they mostly occur at the end of the project, no time is available to complete the learning.

For the ENG.8 Software Testing process, 2 WPs were assessed: 08-21 Software test plan and 15-11 Defect report. Teachers' assessment was Not Achieved for each product; however students' assessment average is 1.56 for the Test Plan (one half Partially and one half Largely) and 1.32 for the Defect Report (two third Partially and one third Largely). Only 2 students self-assessed Not Achieved to both products. The discrepancy might come from the lack of lectures dedicated to testing and from the misunderstanding of the topic. Hence, both WPs are not considered here.

TABLE VI. ENG.8 ASSESSMENT (SELF AND TEACHER)

	Stud.	*Teach.*	*r*
BP1: Develop tests for integrated software product	1.96	1.00	0.53
BP2: Test integrated software product	1.84	1.08	0.27
BP3: Regression test integrated software	1.52	0.56	-0.03

Table 6 presents the correlation coefficient `r` between student and teacher assessment for the ENG.8 Software testing process (*violet*). It relies on 3 Base Practices. Table 6 presents also the average assessment for each assessed item. The overall correlation coefficient is `r = 0.30` and there is little indication for a correlation. Students are not familiar with regression tests and BP3.Regression test integrated software has been assessed by the teacher Not achieved for the half of students and Partially Achieved for the other. If we remove the BP3, we get `r = 0.41`, indicating a possible correlation.

IV. STUDENTS' VIEWPOINT

TABLE VII. STUDENTS' SELF-PERCEPTION ABOUT THE PRACTICUM

The Agenda project	*strg agr*	*agr*	*neu-tral*	*dsgr*	*strg dsgr*
I had the time to learn and do the project.	8	6	3	3	2
I found the project complex.	5	10	5	1	1
I committed to perform the project.	10	10	2	0	0
I found the project realistic.	11	7	2	0	2
I understand relationships between specifications, design, building and tests.	10	6	5	1	0
I had to deepen my knowledge and skills to perform the project.	10	7	2	1	2
My work for the project helped me to understand lectures.	5	3	8	3	3
I used a lot the reviewing facilities.	2	8	7	2	3
I made progress thanks to the reviewing facilities.	3	7	7	2	3
I improved my working methods thanks to the project.	5	10	3	2	2

We were interested to get an unformal feedback. An anonymous questionnaire let students express their opinions about the capstone project, which are presented in Table 7. Although one project objective is to relate to previous lectures and to mobilize knowledge and skills gained during the bachelor studies, it was not effective and rather seen as a new learning experience for the half of students. We were surprised with the poor use and interest for reviewing facilities.

Students' comment. Students appreciated that each project phase has been explained from experience and through examples. Students were convinced of the usefulness of the different phases performed in a software project and that it might be applied to other type of projects. Shared documents could be an alternative to mail exchange and might trigger the use of reviewing facilities that some students misused. Students asked to be exposed to a whole picture of the project at the beginning, not piece by piece. Some students found the work load too heavy and time devoted to the project too short.

V. DISCUSSION AND CONCLUSION

The first research question aims to see if process assessment fosters process learning. The interest of a competency model (process/BPs/WPs) is to supply a reference framework for doing the job. Software professionals may benefit from self-assessment using a competency model in order to record abilities gained through different projects, to store annotations related to new skills, to establish snapshots in order to evaluate and recognize knowledge, skills and experience gained over long periods and in diverse contexts, including in non-formal and informal settings. Students committed to self-assessment but we don't know if they will integrate this reflective practice in their usage or if they did as a part of the capstone project. However, it is a starting point.

The second research question examines how students' self-assessment and external assessment [by a teacher or a third-party] are correlated. This is not true for topics not addressed in the curriculum or unknown by students. For known topics, assessments are correlated roughly for the half of the study population. It might indicate that in a professional setting, where employees are skilled for the required tasks, self-assessment might be a good replacement to external assessment. A 3rd-party assessment did not prove to be useful. Third-party assessment is too harsh and tends to assess almost all students with the same mark.

Acknowledgment

We thank all the students of the 2016-2017 final year of Bachelor in Computer Science to their agreement to participate to this study, and especially Maxens Manach and Killian Monot who collected and anonymized the assessments. We thank Laurence Duval, a teacher that coached and assessed half of the students during the first week.

References

[1] ISO/IEC 15504:2004. Information technology -- Process assessment, Geneva: International Organization for Standardization, 2004.

[2] V. Ribaud et al., "Process Assessment Issues in a Bachelor Capstone Project," in Proceedings of Software Process Education, Training and Professionalism, http://ceur-ws.org/Vol-1368, 2015, pp. 25-33.

[3] V. Ribaud and P. Saliou, "Towards an ability model for software engineering apprenticeship," ITALICS, vol. 6(3), pp. 7-107, 2007.

[4] ACM, 2013 CS Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. http://www.acm.org/education/CS2013-final-report.pdf, last accessed 2017/06/17.